The Claims:

The claims are not amended in this presently-submitted Request For Reconsideration. Thus, the following listing of claims represents the status of the claims following Applicants' previous submission:

Listing of Claims:

Claim 1 (Previously presented). A method for creating an algorithm module that can be used without change in a plurality of frameworks, the method comprising the steps of:

designing the algorithm module in a manner that renders the algorithm module reentrant within a preemptive environment;

coding a plurality of data access instructions of the algorithm module in a manner that renders the algorithm module and the plurality of data access instructions relocatable; and

providing a memory interface within the algorithm module that supports both design-time object instantiation and dynamic object instantiation;

wherein the dynamic object instantiation comprises memory allocation by any framework in the plurality of frameworks in response to memory usage requirements reported to the framework by the algorithm module through the memory interface; and

wherein the design-time object instantiation comprises memory initiation by any framework in the plurality of frameworks in response to memory usage requirements reported to the framework by the algorithm module through the memory interface.

Claim 2 (Previously presented). The method of Claim 1, further comprising the steps of:

conforming to a run-time convention of a selected high level language;

characterizing in the algorithm module whether the algorithm module may or may not be placed in ROM;

prohibiting the algorithm module from direct access to a peripheral device;

packaging the algorithm module in an archive which has a name that follows a uniform naming convention;

naming an algorithm header of the algorithm module using a uniform naming convention; and

naming each external identifier, in a set of external identifiers defined by the algorithm module, according to a uniform naming convention.

Claim 3 (Previously presented). The method of Claim 2, further comprising the step of providing the algorithm module with an initialization function and with a finalization function.

Claim 4 (Previously presented). The method of Claim 3, further comprising the step of providing the algorithm module with a header file that is included in more than one framework in the plurality of frameworks.

Claim 5 (original). The method of Claim 2, further comprising the step of providing a debug variable definition in a header of the algorithm module, wherein the debug variable definition uses the symbol _DEBUG.

Claim 6 (original). The method of Claim 1, further comprising the step of implementing a trace interface as part of the algorithm module.

Claim 7 (Previously presented). A method for converting an existing algorithm to an algorithm module that can be used without change in a plurality of frameworks, the method comprising the steps of:

providing a memory interface for the existing algorithm to form the algorithm module such that the algorithm module supports both design-time object instantiation and dynamic object instantiation;

wherein the dynamic object instantiation comprises memory allocation by any framework in the plurality of frameworks in response to memory usage requirements reported to the framework by the algorithm module through the memory interface; and

wherein the design-time object instantiation comprises memory initiation by any framework in the plurality of frameworks in response to memory usage requirements reported to the framework by the algorithm module through the memory interface;

revising the existing algorithm in a manner that renders the algorithm module reentrant within a preemptive environment; and

verifying that a plurality of data access instructions of the algorithm module are coded in a manner that renders the algorithm module and the plurality of data access instructions relocatable.

Claim 8 (Previously presented). The method of Claim 7, further comprising the steps of:

characterizing in the algorithm module whether the algorithm module may or may not be placed in ROM;

prohibiting the algorithm module from direct access to a peripheral device;

packaging the algorithm module in an archive which has a name that follows a uniform naming convention;

naming an algorithm header of the algorithm module using a uniform naming convention; and

naming each external identifier, in a set of external identifiers defined by the algorithm module, according to a uniform naming convention.

Claim 9 (original). The method of Claim 8, further comprising the step of providing the algorithm module with an initialization function and with a finalization function.

Claim 10 (Previously presented). The method of Claim 9, further comprising the step of providing the algorithm module with a header file header file that is included in more than one framework in the plurality of frameworks.

Claim 11 (original). The method of Claim 10, further comprising the step of providing a debug variable definition in a header of the algorithm module, wherein the debug variable definition uses the symbol _DEBUG.